A new approach to generate hard multiple choice questions from a domain ontology

Vinu E.V[*], P Sreenivasa Kumar

Artificial Intelligence and Databases lab
Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai, India

Abstract

Ontologies contain knowledge in the form of concepts, predicates, instances and their relationships. This knowledge can be exploited by an assessment system in the form of multiple choice questions (MCQs). Most of the existing MCQ generation approaches are limited to questions of type *What is C ?* or *Which of the following is an example of C ?* (where *C* is a concept symbol). Besides the question types, the methods for generating distracting answers (distractors) are to be given much importance. These distractors often determine the quality and difficulty level of a multiple choice question. Currently, there are no systematic methods for generating distractors of a MCQ from an ontology. In this paper, we propose two MCQ generation approaches and the corresponding distractor generating techniques. Our distractor generation techniques, unlike other methods, consider Open World Assumption, so that the generated MCQs will be always precise (ensures falsity of distracting answers). In addition, we present a metric to calculate the difficulty level (a value between 0 and 1) of the generated MCQs. The proposed system is implemented, and experiments on specific ontologies have shown the effectiveness of the approaches. Our results show that the proposed approaches, when used with semantically-rich ontologies, can provide a set of quality MCQs on the domain.

Keywords: Ontologies, semantic web, multiple choice questions, automatic question generation

1. Introduction

Automated assessment systems serve as a method to measure the mastery of learning as well as to provide a fast alternative for large scale assessment. Many tests like TOEFL, IELTS, GRE, GMAT are dependent on online assessment systems to make the assessment task easier. Such systems mainly use multiple choice questions rather than subjective questions for conducting the test.

Using Multiple Choice Questions (MCQs) for assessments have merits and demerits. They are preferred for assessing broad range of knowledge mainly because they require less administrative overhead, and they provide instant feedback to test takers. However, developing effective objective test questions has some disadvantages; it is time consuming and requires domain expertise to generate good quality MCQs. So, there is a requirement for an automated method for MCQ generation from a given knowledge source.

Recently, a handful of studies (Alsubait et al. (2012); Cubric and Tosic (2010); Papasalouros et al. (2008); M.Tosic and M.Cubric (2009); Al-Yahya (2011)) explored the use of structured domain knowledge in the form of ontologies to automatically generate MCQs. This would enable online assessment systems to utilize existing knowledge bases for the assessment of learner's knowledge and skills. But, there are challenges involved in generating MCQs from ontologies. Some of the challenges that we address in this paper are: (i) How to frame interesting and good quality questions from ontologies? (ii) How to generate proper incorrect answers (distractors) for a generated question? (iii) How to control the difficulty level of the generated questions?

Framing interesting and good quality questions. In the literature, there are not many efforts dealing with generating MCQs based on assertional part (ABox) of the ontology. Many of the interesting approaches are

[*]Corresponding author

Email addresses: vinuev@cse.iitm.ac.in (Vinu E.V), psk@cse.iitm.ac.in.com (P Sreenivasa Kumar)

URL: http://aidblab.cse.iitm.ac.in/psk/ (P Sreenivasa Kumar)

based only on terminological facts (concepts and roles describing the structure of the domain) of ontologies. The existing approaches which use ABox axioms generate MCQs of the form: *Which is C?* or *Which of the following is an example of concept C?* (where *C* is a concept symbol). Clearly, these questions are of primitive type and are not much useful to test deeper domain knowledge. The restrictions (existential, universal and cardinality) on classes in ontologies are not utilized by any of the current question generation methods.

Consider a movies related ontology, let `braveHeart` be an instance, and `Movie(braveHeart)`, `ActorsMovie(braveHeart)`, `ActorsMovie` $\equiv$ `Movie` $\sqcap$ $\exists$`isDirectedBy.Actor` be the ontology axioms. Given this knowledge, we can frame a question about the instance `braveHeart` as *Choose the movie which is directed by an actor?* (with *BraveHeart* as the correct answer). Our approach in this paper is an effort in this direction.

Proper distractor generation. Under Closed World Assumption (CWA), we can choose any instance which is different from the instance `braveHeart` as a distracting answer. But, the semantics of Web Ontology Language (OWL) adheres to Open World Assumption (OWA): statements which are not logical consequences of a given knowledge base are not necessarily considered false. Therefore, not all the distracting answers generated under CWA can be considered as a true distractor.

We observed that most of the existing MCQ generation techniques (Al-Yahya (2011)) randomly select instances which do not belong to the class of the correct answer as distractors. The incorrectness of the distractor cannot be ensured by this random selection method. So, there is a need for a systematic method in finding the distractors of a MCQ.

Control the difficulty level of the generated MCQ. MCQs of varying difficulty level are necessary to assess the depth of knowledge of a learner (student). We propose a measure to find out the difficulty level of a MCQ based on the degree to which its distracting answers are related to the right answer. This idea of controlling the difficulty level of a MCQ is motivated by the similarity measure used by Alsubait et al. (2012), to control the difficulty level of analogy type questions. They argue that if the answer option (correct answer) is closer to the question than the distracting answers, the students find the question to be less difficult. Similarly, they expect the question to be difficult, if the question statement is similar to one (or all) of the distractors.

In this paper, we propose two approaches (i) node-label-set based approach (ii) edge-label-set based approach to generate (two) interesting types of MCQs. These two approaches are generic in nature (applicable to any given ontology). We introduce a metric and a method, to control the difficulty level of the generated MCQs, and to choose the most appropriate distracting answers (distractors). We study the feasibility of our approaches by implementing them and by generating MCQs from three example ontologies.

2. Preliminaries

In this section we describe MCQ, the Description Logic (DL) $\mathcal{SHIQ}$ based ontologies ($\mathcal{SHIQ}$ ontologies) and an example ontology (HarryPotter_book ontology).

2.1. Multiple Choice Questions

We adopted a simple and general definition of MCQs (Alsubait et al. (2012)) to explain our approaches and observations.

Definition 1. *A multiple choice question MCQ is a tool that can be used to evaluate whether (or not) a student has attained a certain learning objective. It consists of the following parts:*

(1) **Stem** (S). *The statement that introduces a problem to the student.*
(2) **Choices**. *A set of options* A= $\{A_m \mid 2 \leq m \leq max\}$. *It can be further divided into two sets:*
   a) **Key**. *A set of correct options* K= $\{k_i \mid 1 \leq i < m\}$.
   b) **Distractors**. *A set of incorrect options* D= $\{d_j \mid 1 \leq j <(m-i)\}$, *let* $w = (m-i)$ *be the count of distractors.*

Note : In this paper we assume K as a singleton set.

2.2. $\mathcal{SHIQ}$ DL and $\mathcal{SHIQ}$ Ontologies

The Description logic $\mathcal{SHIQ}$ is based on an extension of the well-known logic $\mathcal{ALC}$ (Schmidt-Schaubßand Smolka (1991)), with added support for role hierarchies, inverse roles, transitive roles, and qualifying number restrictions (Horrocks et al. (2000)).

A $\mathcal{SHIQ}$ ontology is defined using a set of roles, a set of individuals, and a set of concepts (or classes). Roles in a $\mathcal{SHIQ}$ ontology, $\boldsymbol{R}$, can be an atomic role (denoted by R) or an inverse role (denoted by R^-). A set of role inclusion axioms, $R_1 \sqsubseteq R_2$, with $R_1, R_2 \in \boldsymbol{R}$ can be used in defining role hierarchy in an ontology. $\sqsubseteq$ in role inclusion is both reflexive and transitive. A role $R \in \boldsymbol{R}$ is transitive (denoted by Tra(R)), if $R \circ R \sqsubseteq R$ or

$R^- \circ R^- \sqsubseteq R^-$. A role is said to be *simple* if it is neither transitive nor has any transitive subrole (Baader (2007); Horrocks et al. (2000)).

Concepts in a $\mathcal{SHIQ}$ ontology, $\boldsymbol{C}$, can be either an atomic class or a complex class (denoted by C or D). A complex class can be defined using the constructors in Table 1 recursively. In Table 1, $R, S \in \boldsymbol{R}$ with S, a simple role and n is a non-negative integer. The top and bottom concepts ($\top$ and $\bot$ respectively) can be viewed as $(A \sqcup \neg A)$ and $(A \sqcap \neg A)$ respectively.

A $\mathcal{SHIQ}$ ontology is an union of terminology knowledge (called TBox), representing general knowledge of a specific domain, and assertional knowledge (called ABox), representing particular state of the terminology. The names and syntaxes of the axioms in $\mathcal{SHIQ}$ TBox and ABox are listed in Table 2.

Table 1: Syntaxes of $\mathcal{SHIQ}$ concept types

Name	Syntax
atomic concept	A
top concept	$\top$
bottom concept	$\bot$
negation	$\neg C$
conjunction	$C \sqcap D$
disjunction	$C \sqcup D$
existential restriction	$\exists R.C$
universal restriction	$\forall R.C$
min cardinality	$\geq nS.C$
max cardinality	$\leq nS.C$

Table 2: Basic axiom syntaxes of $\mathcal{SHIQ}$ ontology

	Name	Syntax
	role hierarchy	$R \sqsubseteq S$
	role transitivity	Tra(R)
TBox	concept inclusion	$C \sqsubseteq D$
	concept equality	$C \equiv D$
	concept assertion	$C(a)$
ABox	role assertion	$R(a, b)$
	inequality assertion	$a \not\approx b$

Note that explicit assertion of $a \not\approx b$ is supported in $\mathcal{SHIQ}$, but, explicit assertion of equality (represented as $a \approx b$) is not supported, since $\{a\} \equiv \{b\}$ is illegal in $\mathcal{SHIQ}$ (Baader (2007); Hitzler et al. (2009)).

2.3. *HarryPotter_book Ontology*

We use a synthetic ontology called HarryPotter_book ontology (O) as an example ontology throughout this paper. Table 3 and Table 4 contain the TBox and ABox axioms of the ontology respectively.

Some of the explicit knowledge presented in this ontology are explained below. Harry Potter, Hermione, Ron Weasley, Draco Malfoy and Neville Longbottom are students of Hogwarts School. All Hogwarts students should have exactly one Creature as Pet (axiom (1) of Table 3). Owl, Toad, Cat and Rat are the common pets (they are disjoint classes) (axioms (20-25)).

Table 4: The ABox portion of the HarryPotter_book ontology

`DrumstrangStud(viktorKrum)`
`HogStudent(harryPotter)`
`HogStudent(hermione)`
`HogStudent(nevilleLbottom)`
`HogStudent(dracoMalfoy)`
`HogStudent(ronWeasley)`
`Gryffindor(ronWeasley)`
`Gryffindor(harryPotter)`
`Gryffindor(hermione)`
`Gryffindor(nevilleLbottom)`
`Muggle(hermione)`
`Pureblood(dracoMalfoy)`
`Pureblood(ronWeasley)`
`Halfblood(harryPotter)`
`Weasley(ronWeasley)`
`∃hasPet.Cat(hermione)`
`∃hasPet.Rat(ronWeasley)`
`∃hasPet.Owl(harryPotter)`
`∃hasPet.Toad(nevilleLbottom)`
`∃hasPet.Toad(dracoMalfoy)`
`Toad(trevor)`
`Owl(hedwig)`
`Owl(errol)`
`Cat(crookshanks)`
`Rat(scrabber)`
`Slytherin(dracoMalfoy)`
`Slytherin(tomRiddle)`
`hasPet(harryPotter, hedwig)`
`hasPet(nevilleLbottom,trevor)`
`hasPet(ronWeasley,scrabber)`
`hasPet(hermione,crookshanks)`
`Wizard(viktorKrum)`
`Wizard(harryPotter)`
`Wizard(dracoMalfoy)`
`Wizard(tomRiddle)`
`Wizard(nevilleLbottom)`
`knows(nevilleLbottom,harryPotter)`
`knows(harryPotter,tomRiddle)`
`knows(harryPotter,hermione)`
`knows(dracoMalfoy,harryPotter)`
`hasFriend(harryPotter,hermione)`
`hasFriend(harryPotter,ronWeasley)`
`hasHelped(harryPotter,hermione)`
`errol ≉ hedwig`

We have classes like Muggles, Human, Wizard, Pureblood, Halfblood etc in our ontology. Muggles are not Wizards (axiom (13)). All Muggles are Humans (axiom (9)). Pureblood, Halfblood and Muggle are disjoint classes (axioms (16-18)). Hogwarts students belong to either Gryffindor house or Slytherin house (axiom (11)). Harry Potter owns an owl named Hedwig as his pet. Ron Weasley's pet Scrabber is a rat. A toad named Trevor is

Table 3: The TBox axioms of HarryPotter_book ontology

(1) $\mathtt{HogStudent} \equiv \mathtt{Student} \sqcap \forall\mathtt{hasPet.Creature} \sqcap \exists\mathtt{hasPet.Pet} \sqcap \leq 1\mathtt{hasPet.Creature}$

(2) $\mathtt{Pet} \equiv \mathtt{Creature} \sqcap \forall\mathtt{isPetOf.HogStudent} \sqcap \exists\mathtt{isPetOf.HogStudent}$

(3) $\mathtt{HogStudent} \sqsubseteq \mathtt{Student}$ (4) $\mathtt{Student} \sqsubseteq \mathtt{Human}$ (5) $\mathtt{Owl} \sqsubseteq \mathtt{Pet}$ (6) $\mathtt{Rat} \sqsubseteq \mathtt{Pet}$

(7) $\mathtt{Toad} \sqsubseteq \mathtt{Pet}$ (8) $\mathtt{Pet} \sqsubseteq \mathtt{Creature}$ (9) $\mathtt{Muggle} \sqsubseteq \mathtt{Human}$ (10) $\mathtt{Cat} \sqsubseteq \mathtt{Pet}$

(11) $\mathtt{HogStudent} \sqsubseteq \mathtt{Gryffindor} \sqcup \mathtt{Slytherin}$ (12) $\mathtt{DrumstrangStud} \sqcap \mathtt{HogStudent} \sqsubseteq \bot$

(13) $\mathtt{Muggle} \sqcap \mathtt{Wizard} \sqsubseteq \bot$ (14) $\mathtt{Pet} \sqcap \mathtt{Student} \sqsubseteq \bot$

(15) $\mathtt{Pet} \sqcap \mathtt{Human} \sqsubseteq \bot$ (16) $\mathtt{Halfblood} \sqcap \mathtt{Muggle} \sqsubseteq \bot$ (17) $\mathtt{Pureblood} \sqcap \mathtt{Muggle} \sqsubseteq \bot$

(18) $\mathtt{Pureblood} \sqcap \mathtt{Halfblood} \sqsubseteq \bot$ (19) $\mathtt{Gryffindor} \sqcap \mathtt{Slytherin} \sqsubseteq \bot$

(20) $\mathtt{Owl} \sqcap \mathtt{Toad} \sqsubseteq \bot$ (21) $\mathtt{Owl} \sqcap \mathtt{Rat} \sqsubseteq \bot$ (22) $\mathtt{Owl} \sqcap \mathtt{Cat} \sqsubseteq \bot$

(23) $\mathtt{Toad} \sqcap \mathtt{Rat} \sqsubseteq \bot$ (24) $\mathtt{Toad} \sqcap \mathtt{Cat} \sqsubseteq \bot$ (25) $\mathtt{Rat} \sqcap \mathtt{Cat} \sqsubseteq \bot$

(26) $\mathtt{hasFriend} \sqsubseteq \mathtt{knows}$ (27) $\mathtt{hasHelped} \sqsubseteq \mathtt{knows}$ (28) $\mathtt{hasFriend} \sqsubseteq \mathtt{knows}$

(29) $\mathtt{hasPet} \equiv \mathtt{isPetOf}^{-}$

Neville's pet. Hermione's pet is Crookshanks, a cat.

2.4. Label-sets

Our two MCQ generation approaches utilize the *label-set* of individual instances (*node-label-set*) and the *label-set* of pairs of instances (*edge-label-set*) respectively.

Node-label-set. The label-set of an instance is the set which contains the class expressions and (existential, universal and cardinality) restrictions satisfied by that instance. It is represented by $\mathcal{L}(x)$, where x is an instance. For example, the node-label-set of the instance `harryPotter` from Ontology O is given by: $\mathcal{L}$(`harryPotter`) = { `HogStudent, Student, Human, Wizard, Halfblood, Gryffindor,` $\exists$`hasPet.Pet,` $\exists$`hasPet.Owl,` $\forall$`hasPet.Creature,` $\leq$`1hasPet.Creature` }.

Edge-label-set. The label-set of a pair of instances (x, y) is the set that contains all the property relationship (role names) from the first instance to the second instance. It is represented by $\mathcal{L}(x, y)$. From our example ontology O, edge-label-set of the pair (`harryPotter,hermione`) is given by: $\mathcal{L}$(`harryPotter,hermione`) = { `hasFriend, hasHelped, knows` }.

2.5. Label-set Generation techniques

Node-label-set Generation. We generate the label-set of an instance (x) from an ontology O by first creating the corresponding inferred ontology O'(using a reasoner). From the inferred ontology O', the set of all classes that have x as an instance can be easily obtained by a simple SPARQL query:

```
@prefix ex:<http://example.org>
 select distinct ?concept
 where {ex:x a ?concept}
```

For example, we generate the label set of the instance `harrypotter` ($\mathcal{L}$(`harryPotter`)) of ontology O by running the SPARQL query: `select distinct ?concept where { hp:harryPotter a ?concept }` on the inferred ontology O' ($O \models O'$). We get $\mathcal{L}$(`harryPotter`) as { `HogStudent, Student, Human, Wizard, Halfblood, Gryffindor` }. We maintain the disjoint classes of each of the named classes in the label-set $\mathcal{L}(x)$ as a set called *Disjoint*($\mathcal{L}(x)$). For example, *Disjoint*($\mathcal{L}$(`harryPotter`)) = { `Pet, Owl, Rat, Toad, Cat, Muggle, Pureblood, Slytherin` }. We later use *Disjoint* sets of label-sets in Section 3.1.4 for generating distractors.

In order to get the restrictions satisfied by an instance, we access the restrictions on the class definitions and super class expressions of the concepts (*named concepts*) that are present in its label set. We then add the existential, universal and cardinality restrictions on the right hand side of these expressions to the label-set of the instance.

Continuing with our example of the instance `harryPotter`, enrichment of the label set is done by obtaining (i) existential, universal and cardinality restrictions on each of the atomic classes in the label set (i.e., the set {$\exists$`hasPet.Pet,` $\forall$`hasPet.Creature,` $\leq$`1hasPet.Creature`}) and (ii) other restrictions on the instance (i.e., {$\exists$`hasPet.Owl`}). We can use

Jena Ontology API[1] or OWL API[2] for extracting the restriction information from the ontology file. Finally, we get $\mathcal{L}$(`harryPotter`) as { `HogStudent, Student, Human, Wizard, Halfblood, Gryffindor, ∃hasPet.Pet, ∃hasPet.Owl, ∀hasPet.Creature, ≤1hasPet.Creature` }. The node-label-sets of other instances in the HarryPotter_book ontology are given in Table 5.

Edge-label-set Generation. The edge-label-set of a pair of instances (x, y) can be easily generated using the SPARQL query:

```
@prefix ex:<http://example.org>
 select distinct ?predicate
 where { ex:x ?predicate ex:y }
```

We get $\mathcal{L}$(`harryPotter,hermione`) = { `hasFriend, hasHelped, knows` } from the above query by substituting x = `harryPotter` and y = `hermione`. Other necessary edge-label-sets used in this paper are given in Table 8.

3. Proposed MCQ Generation Approaches

Once we get the label-set of all instances (node-label-set) and label-set of all the pairs of instances (edge-label-set) in the knowledge base, we can generate the MCQs based on the following two approaches.

3.1. Node-label-set Based Approach

In this approach, we frame a question based on the label-set of an instance (with that instance as the key). But a label-set needs to be processed for making it suitable to frame the stem and for relating it to other node-label-sets to generate distractors. So, we introduce a procedure called Label-set-Reduction and set of reduction rules, to process the label-set. We also discuss our distractor generating method, and the metrics to calculate the difficulty of the generated MCQ in detail below.

3.1.1. Label-set-Reduction

Consider the node-label-sets in the Table 5, we find that the generation of stem using those label-sets is a difficult task, since the cardinality of the set is large and many of the terms implicitly carry same information. For example, the cardinality of the $\mathcal{L}$(`harryPotter`) is 10, and the terms like `HogStudent` in label-set implicitly states `harryPotter` is a member of the class `Student`. Therefore, the set {`HogStudent`, `Student`} can be reduced to {`HogStudent`}.

We introduce a cardinality reduction procedure called Label-set-Reduction to minimize the cardinality of the label set to make it easier to frame the stem. This cardinality reduction (not considering terms with quantifiers or cardinality restriction) is done by finding out the relationship between concepts in the set and choosing the most specific concepts alone. For example, if the label-set $\mathcal{L}(x)$ is $\{R, S, T\}$ and if $R \sqsubseteq S$, then after Label-set-Reduction the set becomes $\{R, T\}$. Therefore, from the label-set of `harryPotter`, the concepts `Human` and `Student` can be removed because of the TBox axioms (4) and (3) (in the Table 3) respectively. After reducing the concept hierarchy, the label-set of `harryPotter` becomes { `HogStudent, Wizard, Halfblood, Gryffindor, ∃hasPet.Pet, ∃hasPet.Owl, ∀hasPet.Creature, ≤1hasPet.Creature` }. We call this new node-label-set as *Reduced-node-label-set* ($\mathcal{R}nls$, for short) of `harryPotter` (denoted as, $\mathcal{R}(\mathcal{L}$(`harryPotter`))).

3.1.2. Reduction rules

Since we are interested in $\mathcal{SHIQ}$ ontologies, in addition to the quantified (universal and existential) restrictions we can expect terms with qualified cardinality restriction in the *node-label-set*. To extend the Label-set-Reduction to incorporate quantified and qualified cardinality expressions, we have formulated a set of *reduction rules* (Table 6). The table shows the reduction of quantified and qualified cardinality restrictions on an instance by considering a pair of restrictions at a time. Pairs that are not shown in Table 6 cannot be reduced and hence they are used as they are in the Reduced-node-label-set ($\mathcal{R}nls$).

The reduction rules are applied in five phases. Each phase handles the reduction of carefully chosen restriction patterns whose resulting patterns are being used for further reduction in the next phase. Phase 1: This phase considers restriction of the form $\exists R.C$ alone, and rule(1) is applied to all the possible pairs in the $\mathcal{R}nls$. Phase 2: This phase handles restriction of the form $\forall R.C$, and rules(2-4) are applied in a similar way to the resulting set of phase 1. Phase 3: Rule(5-11) to the applicable pairs in the result set of the previous phase. Phase 4: Use Rule(12-13). Phase 5: Rule(14-15) to all possible pairs in the result set. The reasons behind reducing the label-set in phases are, (1) the restrictions of the same type can be reduced easily, and (2) all the possible reductions can be done by the rules

[1]`http://jena.apache.org`
[2]`http://owlapi.sourceforge.net`

Table 5: Label-set of the instances in the example ontology

$\mathcal{L}$(harryPotter)	{ HogStudent, Student, Human, Wizard, HalfBlood, Gryffindor, ∃hasPet.Pet, ∃hasPet.Owl, ∀hasPet.Creature, ≤1hasPet.Creature }
$\mathcal{L}$(ronWeasley)	{ HogStudent, Student, Human, Weasley, Gryffindor, Pureblood, ∃hasPet.Pet, ∃hasPet.Rat, ∀hasPet.Creature, ≤1hasPet.Creature }
$\mathcal{L}$(hermione)	{ HogStudent, Student, Human, Muggle, Gryffindor, ∃hasPet.Pet, ∃hasPet.Cat, ∀hasPet.Creature, ≤1hasPet.Creature }
$\mathcal{L}$(dracoMalfoy)	{ HogStudent, Student, Human, Slytherin, Pureblood, ∃hasPet.Pet, ∃hasPet.Toad, ∀hasPet.Creature, ≤1hasPet.Creature }
$\mathcal{L}$(hedwig)	{ Owl, Pet, Creature, ∀isPetOf.HogStudent, ∃isPetOf.HogStudent }
$\mathcal{L}$(errol)	{ Owl, Pet, Creature }
$\mathcal{L}$(scrabber)	{ Rat, Pet, Creature, ∀isPetOf.HogStudent, ∃isPetOf.HogStudent }
$\mathcal{L}$(trevor)	{ Toad, Pet, Creature, ∀isPetOf.HogStudent, ∃isPetOf.HogStudent}
$\mathcal{L}$(crookshanks)	{ Cat, Pet, Creature, ∀isPetOf.HogStudent, ∃isPetOf.HogStudent }
$\mathcal{L}$(nevilleLbottom)	{ HogStudent, Student, Wizard, Gryffindor, ∃hasPet.Pet, ∃hasPet.Toad, ∀hasPet.Creature, ≤1hasPet.Creature }
$\mathcal{L}$(viktorKrum)	{ Wizard, DrumstrangStud }
$\mathcal{L}$(tomRiddle)	{ Wizard, Slytherin }

given in Table 6. To illustrate our rationale, consider an instance with label-set { ∃hasPet.Creature, ∀hasPet.Creature, ∀hasPet.Owl }. If we are not following any particular order for reduction, for n number of terms, there are $^{n}C_2$ ways for choosing a pair. In our example, we can take any of the 3 pairs (∃hasPet.Creature, ∀hasPet.Creature), (∀hasPet.Creature, ∀hasPet.Owl) and (∃hasPet.Creature, ∀hasPet.Owl) for starting the reduction. Figure 1 shows the reduction w.r.t. each of the above 3 pairs. In the figure, (1) and (3) requires additional rule other than the basic reduction rules in Table 6, to proceed to a single reduced term, and (2) follows our phase based reduction and requires no addition rule.

We introduce two notations ∃! and $\mathfrak{I}$ in our reduction rules in Table 6. ∃! in $\exists! R.C$ is equivalent to $\exists R.C \sqcap \leq 1 R.C$. For example, ∃!hasPet.Owl denotes that there exist exactly one hasPet relation to the class Owl. $\mathfrak{I}$ in $\mathfrak{I}R.C$ is used to represent the pair $(\exists R.C, \forall R.C)$. i.e, $\mathfrak{I}R.C \equiv \exists R.C \sqcap \forall R.C$ (i.e., ∃isPetOf.HogStudent and ∀isPetOf.HogStudent is represented as $\mathfrak{I}$isPetOf.HogStudent). If the restrictions like $\exists R.C$ and $\forall R.C$ appear together in a label-set, the former guarantees the presence of an edge R and makes it necessary that the latter should be satisfied by the condition other than vacuously true case. This observation is useful when we frame the question statement (stem) of a MCQ.

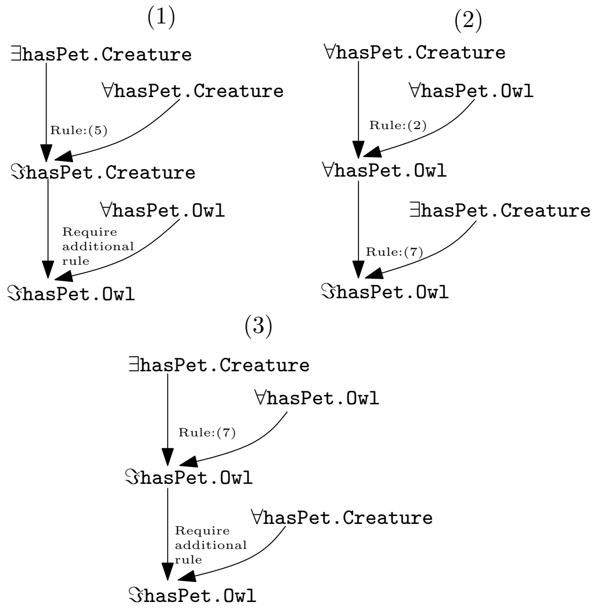

Figure 1: Applying reduction rules to pairs of restrictions taken in three different orders. (1) and (3) requires additional rule to proceed to a single reduced term, and (2) follows our phase based reduction order and requires no addition rule.

A diagrammatic representation of the application of the reduction rules on the restrictions in $\mathcal{R}(\mathcal{L}$(harryPotter)) is shown in Figure 2. Phase 1 and phase 2 rules are not applicable to the restrictions in $\mathcal{R}(\mathcal{L}$(harryPotter)). So, the diagram shows the re-

Table 6: Reduction rules

No.	*Restriction*1	*Restriction*2	*Condition*	*Result*
1	$\exists R_u.C_u$	$\exists R_v.C_v$	$C_u \sqsubseteq C_v$ & $R_u \sqsubseteq R_v$	$\exists R_u.C_u$
2	$\forall R_u.C_u$	$\forall R_v.C_v$	$C_u \sqsubseteq C_v$ & $R_v \sqsubseteq R_u$	$\forall R_u.C_u$
3	$\forall R_u.C_u$	$\forall R_v.C_v$	$C_v \sqsubseteq C_u$ & $R_v = R_u$	$\forall R_v.C_v$
4	$\forall R_u.C_u$	$\forall R_v.C_v$	$C_v \sqsubseteq C_u$ & $R_u \sqsubset R_v$	$\forall R_u.C_v, \forall R_v.C_v$
5	$\exists R_u.C_u$	$\forall R_v.C_v$	$R_u = R_v$ & $C_u = C_v$	$\Im R_u.C_u$
6	$\forall R_u.C_u$	$\exists R_v.C_v$	$C_v \sqsubset C_u$ & $R_u \sqsubseteq R_v$	$\Im R_u.C_u, \exists R_v.C_v$
7	$\forall R_u.C_u$	$\exists R_v.C_v$	$C_u \sqsubseteq C_v$ & $R_v \sqsubseteq R_u$	$\Im R_u.C_u, \Im R_v.C_u$
8	$\forall R_u.C_u$	$\exists R_v.C_v$	$C_u \sqsubseteq C_v$ & $R_u \sqsubset R_v$	$\forall R_u.C_u, \exists R_v.C_v$
9	$\exists R_u.C_u$	$\geq nR_v.C_v$	$C_v \sqsubseteq C_u$ & $R_v \sqsubseteq R_u$ & $n \geq 1$	$\geq nR_v.C_v$
10	$\geq n_uR_u.C_u$	$\geq n_vR_v.C_v$	$C_u \sqsubseteq C_v$ & $R_u \sqsubseteq R_v$ & $n_u \geq n_v$	$\geq n_uR_u.C_u$
11	$(\geq n)\exists R_u.C_u$	$\leq nR_v.C_v$	$C_u \sqsubseteq C_v$ & $R_u \sqsubseteq R_v$ & $n = 1$	$\exists ! R_u.C_u$
12	$(\Im)\exists R_u.C_u$	$\exists ! R_v.C_v$	$C_u \sqsubseteq C_v$ & $R_u \sqsubseteq R_v$	$\exists ! R_u.C_u$
13	$(\Im)\exists R_u.C_u$	$\exists ! R_v.C_v$	$C_v \sqsubseteq C_u$ & $R_u \sqsubseteq R_v$	$\exists ! R_u.C_v$
14	$\exists ! R_u.C_u$	$\exists ! R_v.C_v$	$C_u \sqsubseteq C_v$ & $R_u \sqsubseteq R_v$	$\exists ! R_u.C_u, \exists ! R_v.C_u$
15	$\exists ! R_u.C_u$	$\exists ! R_v.C_v$	$C_v \sqsubseteq C_u$ & $R_u \sqsubseteq R_v$	$\exists ! R_u.C_v, \exists ! R_v.C_v$

duction phase 3 to 5. In the figure, the pairs of restrictions which satisfy the conditions in Table 6 are reduced based on the corresponding rules. The directed lines in phase 3 (in the figure) are paired such that each pair represents the applicability of a rule. In phase 4, the restrictions are connected pairwise and the arrow head points to the resultant restriction. In phase 5, rules (14-15) are repeatedly applied to find the final result. From here on, $\mathcal{R}nls$ denotes the label-set after applying Label-set-Reduction and the reduction rules.

3.1.3. Stem-set Generation Heuristics

Stem-sets are the sets which are used for framing the question statements of MCQs. Reduced-node-label-set of an instance is normally used as the stem-set for framing a question. Example 1 shows the stem generated from $\mathcal{R}nls$ of `harryPotter`. But there are cases where we cannot use a $\mathcal{R}nls$ as it is for a stem generation.

Case 1. A Reduced-node-label-set of an instance can contain restrictions of the form $\forall R.C$ (universal quantifier). These universal quantifiers in a $\mathcal{R}nls$ are restrictions which may be satisfied by the instance under *vacuously true condition* (otherwise, they would have been reduced by the reduction rules in Table 6). We observed that using such restrictions in framing the stem affects the clarity of the question. For example, consider the Reduced-node-label-set of two instances namely `john` and `bob`. Let their Reduced-node-label-sets be $\mathcal{R}(\mathcal{L}($`john`$))$ = { `Engineer`, ∀`hasChild.Doctor` } and $\mathcal{R}(\mathcal{L}($`bob`$))$ = { `Engineer`, ℑ`hasChild.Doctor` } respectively. The question framed from the latter $\mathcal{R}nls$ is *Choose an Engineer, having only Doctor as child.* This stem does not mislead the person undertaking the test, since the answer to the question can be unambiguously chosen as `bob`. This unambiguous selection is because of the fact that the linguistic interpretation and the logical statement (the $\mathcal{R}nls$) of the stem remains the same. But the question generated from the former Reduced-node-label-set misleads the test taker, since it asks to choose an engineer whose children are all doctors. But, there can be a case that `john` satisfied the restriction ∀`hasChild.Doctor` only by vacuously true manner. In this case, the linguistic interpretation of the logical statement under vacuously true case may confuse the test taker. Therefore, we made a design decision to remove such universal restrictions that may be satisfied by vacuously true condition from the $\mathcal{R}nls$ to generate the corresponding stem-set. So, the stem-set of the instance `john` becomes {`Engineer`}, and the stem-set of the instance `bob` remains the same as that of its $\mathcal{R}nls$.

Case 2. The restrictions which contain bottom or top concept can be removed from the $\mathcal{R}nls$ while generating stem-sets. For example, terms of the form ∃`hasChild.`⊤, ∃`hasPet.`⊥ can be removed from the Reduced-node-label-set. The terms with bottom or top concept add vagueness to the stems of the generated multiple choice questions.

A formal definition of the MCQ based on node-label-

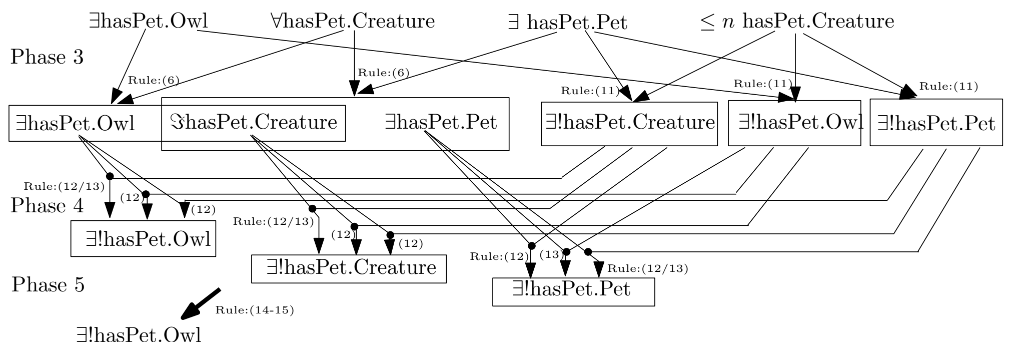

Figure 2: Application of reduction rules on the restrictions in $\mathcal{R}(\mathcal{L}($`harryPotter`$))$

Table 7: Hierarchy Reduced node-label-sets of the ontology *O*

$\mathcal{R}(\mathcal{L}($harryPotter$))$	{ HogStudent, Wizard, HalfBlood, Gryffindor, ∃!hasPet.Owl }
$\mathcal{R}(\mathcal{L}($ronWeasley$))$	{ HogStudent, Weasley, Gryffindor, Pureblood, ∃!hasPet.Rat }
$\mathcal{R}(\mathcal{L}($hermione$))$	{ HogStudent, Muggle, Gryffindor, ∃!hasPet.Cat }
$\mathcal{R}(\mathcal{L}($dracoMalfoy$))$	{ HogStudent, Slytherin, Pureblood, ∃!hasPet.Toad }
$\mathcal{R}(\mathcal{L}($hedwig$))$	{ Owl, ℑisPetOf.HogStudent }
$\mathcal{R}(\mathcal{L}($errol$))$	{ Owl }
$\mathcal{R}(\mathcal{L}($scrabber$))$	{ Rat, ℑisPetOf.HogStudent }
$\mathcal{R}(\mathcal{L}($trevor$))$	{ Toad, ℑisPetOf.HogStudent }
$\mathcal{R}(\mathcal{L}($crookshanks$))$	{ Cat, ℑisPetOf.HogStudent }
$\mathcal{R}(\mathcal{L}($nevilleLbottom$))$	{ HogStudent, Wizard, ∃!hasPet.Toad }
$\mathcal{R}(\mathcal{L}($viktorKrum$))$	{ Wizard, DrumstrangStud }
$\mathcal{R}(\mathcal{L}($tomRiddle$))$	{ Wizard, Slytherin }

set is given in the Definition 2. The discussion on how to convert a stem-set to a grammatically correct question statement is out of the scope of this paper.

3.1.4. Distractor Generation Procedure

In the last section, we have seen that the MCQs are generated using the stem-set of each of the instances in a given ontology. The instance corresponding to the stem-set will become the correct answer. Distractors are the set of incorrect options of a MCQ. Given an instance x (the correct answer of a MCQ), we use its Reduced-node-label-set ($\mathcal{R}(\mathcal{L}(x))$) to identify the set of all instances which can be considered as distractors ($D_{complete}$). This distractor selection is based on the following condition.

$$\mathcal{R}(\mathcal{L}(x)) \;\cap\; Disjoint\ (\mathcal{L}(d)) \;\neq\; \phi \qquad (1)$$

In Condition (1), d is an instance other than x. *Disjoint* set is the set which is obtained by taking the union of the set of disjoint classes of each of the *named classes* in the label-set of an instance. The above condition guarantees that there will be at least one *named concept* in the stem forming set (stem-set) which is not satisfied by the distractor instance. This condition also prevents the possibility of a distractor becoming a right answer under open world assumption. We call the instance d that satisfies Condition (1) as a candidate for the distractor of the generated MCQ. The trivial algorithm for distractor generation from a given instance is given below (Algorithm 1). In Example 1, to find the distractors of the correct answer instance `harryPotter`, we considered instances like `hermione`, `hedwig` etc. which satisfies the condition (1). In the below illustration, we show that the two instances, `hermione` and `hedwig` are semantically different from `harryPotter`, and it can be guaranteed that the distractor instances are disjointed with at least one named concept which is used as a condition in the stem.

Disjoint $(\mathcal{L}($hermione$))$ = { Pet, Owl, Rat, Toad, Cat, Wizard, Halfblood, Pureblood, Slytherin }

Disjoint $(\mathcal{L}($hedwig$))$ = { Rat, Toad, Cat, Muggle, `Gryffindor, Slytherin, Student, HogStudent, Human` }

$\mathcal{R}(\mathcal{L}($`harryPotter`$)) \cap$ *Disjoint* $(\mathcal{L}($`hermione`$))$ = { `Halfblood, Wizard` }

$\mathcal{R}(\mathcal{L}($`harryPotter`$)) \cap$ *Disjoint* $(\mathcal{L}($`hedwig`$))$ = { `HogStudent, Gryffindor` }

Algorithm 1 Distractor generating algorithm

```
1: Input: $x$ (correct answer instance)
2: Output: $D_{complete}$ (List of possible distractors)
3: InstanceList $D_{complete} \leftarrow \phi$
4: for all instance $d$ in $O$ do
5:     if ( $(\mathcal{R}(\mathcal{L}(x)) \cap Disjoint(\mathcal{L}(d)) \neq \phi$ ) then
6:         Add $d$ to $D_{complete}$
7:     end if
8: end for
9: return $D_{complete}$
```

By Definition 1, we need at least w number of distractors in the $D_{complete}$ set corresponding to the correct answer instance x (or corresponding to $\mathcal{R}nls$ of x). Those $\mathcal{R}(\mathcal{L}(x))$ that satisfies this condition can be used for generating *stem-sets*. In other words, for a stem-set (S), there should be at least w number of distractors in its $D_{complete}$ set.

Definition 2. ***Type 1:*** *Let* Q_1 *be the multiple choice question with stem-set* S*, generated from* $\mathcal{R}(\mathcal{L}(x))$*, where* $\mathcal{R}(\mathcal{L}(x))$ *is the Reduced-node-label-set of the instance (with label) x. Key of* Q *is defined as* K= $\{x\}$ *and distractors is defined as* D = $\{\ y \mid y \neq x$ *and* $(\ Disjoint(\mathcal{L}(y)) \cap S\ ) \neq \phi\ \}$.

We introduce a similarity measure called Label-set similarity ratio (LSR) to find out how closely an instance is related to another instances w.r.t. their $\mathcal{R}nls$. In the next section, we use this measure to identify the distractors (in $D_{complete}$) which are closely related to the correct answer.

Definition 3. *Label-set Similarity ratio (LSR). Given two Reduced-node-label-sets U, V; Label-set similarity ratio of U and V (denoted by, LSR(U,V)) is the ratio of number of elements in U which can be associated to some elements in V, to the total number of elements in U.*

$$LSR(U,V) = \frac{|\{\ u \mid u \in U \text{ and } \exists\ v \in V \text{ st. } u \rightsquigarrow v\}|}{|U|}$$

Association between an element $u \in U$ to $v \in V$ (denoted by $u \rightsquigarrow v$) is based on the conditions mentioned below.

Since we are interested in the Reduced-node-label-sets (U and V) of a $\mathcal{SHIQ}$ ontology, the elements of U (or V) can be a concept name or a term of the form $\exists R.C$ or $\Im R.C$ or $\leq nR.C$ or $\geq nR.C$ or $\exists!R.C$.

We associate $u \in U$ to $v \in V$ based on the following conditions:

1. A concept $u \in U$ is associated to a concept $v \in V$, if $u \sqsubseteq v$ (*concept subsumption*).
2. If $u \in U$ is of the form $\exists R_u.C_u$, we associate it to a v ($\in V$) of the form $\exists R_v.C_v$ or $\Im R_v.C_v$ or $\exists!R_v.C_v$ such that $R_u \sqsubseteq R_v$ and $C_u \sqsubseteq C_v$.
3. The terms of the form $\Im R_u.C_u$ in U are associated to the terms of the form $\Im R_v.C_v$ or $\exists!R_v.C_v$ in V with $R_u \sqsubseteq R_v$ and $C_u \sqsubseteq C_v$.
4. If u is of the form $\leq n_u R_u.C_u$, we associate it to a v of the form $\leq n_v R_v.C_v$, such that $n_v \geq n_u$, $R_u \sqsubseteq R_v$ and $C_u \sqsubseteq C_v$.
5. If u is of the form $\geq n_u R_u.C_u$, we associate it to a v of the form $\geq n_v R_v.C_v$, such that $n_v \leq n_u$, $R_u \sqsubseteq R_v$ and $C_u \sqsubseteq C_v$.
6. If u is of the form $\exists!R_u.C_u$, we associate it to a v of the form $\exists!R_v.C_v$, such that $R_u \sqsubseteq R_v$ and $C_u \sqsubseteq C_v$.

For example, consider the $\mathcal{R}nls$ of `harryPotter` and `hermione` in Table 7. There are two associations from the first label-set to the second. These two associations (shown below) are because of the first association condition (concept subsumption).

$\mathcal{R}(\mathcal{L}($`harryPotter`$))$		$\mathcal{R}(\mathcal{L}($`hermione`$))$
`HogStudent`	$\rightsquigarrow$	`HogStudent`
`Wizard`		`Muggle`
`Halfblood`		
`Gryffindor`	$\rightsquigarrow$	`Gryffindor`
`∃!hasPet.Owl`		`∃!hasPet.Cat`

Let U = $\mathcal{R}(\mathcal{L}($`harryPotter`$))$ and V = $\mathcal{R}(\mathcal{L}($`hermione`$))$; then, |U| = 5 and |V| = 4.

$$\begin{aligned} LSR(U,V) &= {}^{\{\texttt{Gryffindor, HogStudent}\}}/_{|U|} \\ &= \tfrac{2}{5} = 0.4 \end{aligned}$$

3.1.5. Difficulty level of MCQs.

The difficulty level of a generated MCQ can be calculated based on how the $\mathcal{R}nls$ of its distractors are related to the $\mathcal{R}nls$ of the correct answer. If the distractors are closely related to the correct answer, the difficulty level of the MCQ is (intuitively) high. The closeness between the two $\mathcal{R}nls$ (U and V) is measured in terms of their Label-set Similarity ratio.

Definition 4. *Given two $\mathcal{R}nls$ U and V, Closeness is defined as the average value of LSR(U,V) and LSR(V,U).*

$$Closeness(U, V) = \frac{LSR(U, V) + LSR(V, U)}{2} \quad (2)$$

The difficulty level of a MCQ (Q) with s as the correct answer and $\{x_1, x_2 ... x_w\}$ as w (where $w \neq 0$) distractors is defined as:

$$Difficulty(Q(s, x_1, x_2 ... x_w)) = \left(\frac{\sum_{i=1}^{w} Closeness(\mathcal{R}(\mathcal{L}(s)), \mathcal{R}(\mathcal{L}(x_i)))}{w} \right) \quad (3)$$

Consider our previous example, *Closeness* of the Reduced-node-label-sets $\mathcal{R}(\mathcal{L}($`harryPotter`$))$ and $\mathcal{R}(\mathcal{L}(\text{hermione}))$ (U and V respectively) is obtained as follows:

$$LSR(U, V) = 0.4, \ LSR(V, U) = 0.5$$

$$Closeness(U, V) = \frac{0.4 + 0.5}{2} = 0.45$$

Example 1. Choose a `Hogwarts Student`, a `Wizard`, a `Gryffindor` and a `Halfblood`, `having exactly one Owl as Pet.`

	Options	*Closeness*
a)	HarryPotter	1
b)	Hermione	0.45
c)	TomRiddle	0.35
d)	Hedwig	0
	Difficulty	0.266

	Options	*Closeness*
a)	HarryPotter	1
b)	RonWeasley	0.4
c)	Hermione	0.45
d)	Hedwig	0
	Difficulty	0.283

	Options	*Closeness*
a)	HarryPotter	1
b)	DracoMalfoy	0.225
c)	RonWeasley	0.4
d)	Herminone	0.45
	Difficulty	0.358

In the above example, we have shown three sets of options for the MCQ. The difficulty level of the sets are 0.266, 0.283 and 0.358 respectively. Since we assume that the difficulty level of a MCQ is related to the closeness of its options, here, the third set of options makes the MCQ more difficult than the other two sets. Intuitively, the third set of options is indeed the difficult one among all the three sets. This is because, the three distractors DracoMalfoy, RonWeasley and Hermione are closely related to HarryPotter w.r.t. the conditions mentioned in the stem-set. But, in the other sets of options an owl named Hedwig is appearing as a distracting answer. Hedwig, being an odd one among the other options, can be easily ruled down by the person undertaking the test as a wrong answer. Clearly, the presence of Hedwig as an option is reducing the difficulty values of the MCQs which are generated using the first two sets of options.

3.2. Edge-label-set Based MCQ Generation

If an instance is related to another instance by a predicate, we can frame a question of type: <instance_1> <predicate>_____. Even though a correct answer can be easily obtained by identifying the triple patterns in the ontology, the selection of distractors under open world assumption is difficult. Let us consider the edge-label-sets of the pair (`harryPotter`, `ronWeasley`) from Table 8.

$\mathcal{L}$(`harryPotter, ronWeasley`) = { `hasFriend` }

We can generate the stem *HarryPotter has Friend* ______, from the above label-set. Some of the possible options are *a. RonWeasley b. Hermione c. TomRiddle d. Hedwig*. Both options a and b are true, since there are explicit triples, but the options c and d cannot be said as false (they are just not known). Such unknown options which cannot be a distractor can be avoided only by considering those predicates whose range is limited to some classes or a few instances, for stem generation.

In this section, we propose an approach to identify those predicates/properties in the edge-label-set which are favorable in generating a stem. We consider each of the edge-label-sets and identify the predicates which

Table 8: Some edge-label-sets of the ontology O

Edge-label-set		P_1	P_2
$\mathcal{L}$(harryPotter, ronWeasley)	{ hasFriend }	ϕ	ϕ
$\mathcal{L}$(harryPotter, hermione)	{ knows, hasFriend, hasHelped }	ϕ	ϕ
$\mathcal{L}$(harryPotter,hedwig)	{ hasPet }	ϕ	{ hasPet }
$\mathcal{L}$(nevilleLbottom, trevor)	{ hasPet }	ϕ	{ hasPet }
$\mathcal{L}$(ronWeasley, scrabber)	{ hasPet }	ϕ	{ hasPet }
$\mathcal{L}$(hermione, crookshanks)	{ hasPet }	ϕ	{ hasPet }
$\mathcal{L}$(hedwig, harryPotter)	{ isPetOf }	{ isPetOf }	ϕ
$\mathcal{L}$(trevor, nevilleLbottom)	{ isPetOf }	{ isPetOf }	ϕ
$\mathcal{L}$(scrabber, ronWeasley)	{ isPetOf }	{ isPetOf }	ϕ
$\mathcal{L}$(crookshanks, hermione)	{ isPetOf }	{ isPetOf }	ϕ

can be used for generating a stem using the following algorithm (Algorithm 2).

Algorithm 2 Selecting predicates for stem generation

```
1: Input: $\mathcal{L}(a,b)$ (edge-label-set of instances $a$ and $b$)
2: Output: $P_1$, $P_2$ (predicate lists)
3: PredicateList $P_1 \leftarrow \phi$
4: PredicateList $P_2 \leftarrow \phi$
5: for all $R \in \mathcal{L}(a,b)$ do
6:     comment: For some concept $C$.
7:     if ($\Im R.C \in \mathcal{R}(\mathcal{L}(a))$) and ($C \in \mathcal{L}(b)$) then
8:         Add $R$ to $P_1$
9:     end if
10:    if ($\exists! R.C \in \mathcal{R}(\mathcal{L}(a))$) and ($C \in \mathcal{L}(b)$) then
11:        Add $R$ to $P_2$
12:    end if
13: end for
14: return $P_1$ and $P_2$
```

For a given edge-label-set $\mathcal{L}(a,b)$, the algorithm choose the predicates in $\mathcal{L}(a,b)$ one by one (lines 5-13) and checks if they forms a part of the restrictions of the form $\Im R.C$ or $\exists! R.C$ in the $\mathcal{R}nls$ of the instance a. If such a restriction exists and if $C \in \mathcal{L}(b)$, then R can be considered for framing a stem of the form: a R ____.

If the predicate R appears as a part of $\Im R.C$, and $C \in \mathcal{L}(b)$, then we store the predicate R in list P_1 (lines 7-9). Similarly, the predicates which are appearing in the restrictions of the form $\exists! R.C$ are stored in P_2 (lines 10-12). We handle these two lists separately for MCQ generation. The reason for maintaining two lists is explained below. Table 8 shows list of predicates in P_1 and P_2 returned by Algorithm 2, by giving corresponding edge-label-set as input.

3.2.1. MCQ generation from edge-label-set and list P_2

Consider the edge-label-set $\mathcal{L}$(harryPotter, hedwig) = { hasPet }. From Table 7, we get, the restriction $\exists!$hasPet.Owl is an element of $\mathcal{L}$(harryPotter). Since the class Owl in the restriction is contained in $\mathcal{L}$(hedwig), we can frame a stem based on $\mathcal{L}$(harryPotter, hedwig) and hasPet. Example 2 shows the generated MCQ.

According to Algorithm 2, given $\mathcal{L}$(harryPotter, hedwig) as input, we store the predicate hasPet in the list P_2. Significance of this list is that all the predicates of the input edge-label-set which are having a strict predicate restriction are stored in it. For example, *HarryPotter* has exactly one pet which is an owl named *Hedwig*. So intuitively, we only need to choose all other instances (including other owls) which are different from hedwig as distractors. Therefore, we can use the algorithm discussed in the previous section (Algorithm 1) for finding the distractors.

Given an edge-label-set $\mathcal{L}(a,b)$ with $P_2 \neq \phi$, we generate distractors by giving a as the input to Algorithm 1. The explicit inequalities of individuals in the ontology can be also utilized in this case as a condition for choosing the distractor set. Therefore, Condition (1) which is used in Algorithm 1 (line 5) can be modified as per the following equation:

$$D_{complete} = \{ d_i \mid \mathcal{R}(\mathcal{L}(x)) \cap Disjoint\ (\mathcal{L}(d_i)) \neq \phi \quad or \quad x \not\approx d_i \} \quad (4)$$

The relevance of this distractor generation method can be explained by the following simple example. In the above discussion of Example 2, we have seen that hedwig is an Owl; intuitively, those Owl instances which are different from hedwig form the most apt distractors. Our algorithm returns such instances also in the complete distractor list, even if the inequalities of the instances are not explicitly given in the ontology. To elaborate it further, assume that the `Owl` class has two disjoint subclasses: `BlackOwl` and `WhiteOwl`. `hedwig` belongs to `BlackOwl`, and another instance `browner` belongs to `WhiteOwl`. Also, assume that the axiom `hedwig` $\not\approx$ `browner`, is not present in the ontology. Since the restriction $\exists!$`hasPet.Owl` is on `Owl`, the obvious way to choose the complete distractors is by finding all instances which belong to $\neg$`Owl`. But, in Algorithm 1, we consider an instance as a distractor if $\mathcal{R}(\mathcal{L}(x)) \cap Disjoint(\mathcal{L}(d)) \neq \phi$. Therefore, the disjointness of classes `BlackOwl` and `WhiteOwl` is enough to make `browner` a distractor.

Example 2. HarryPotter has Pet ______.

	Options	*Closeness*
a)	Hedwig	1
b)	Crookshanks	0.5
c)	Errol	0.75
d)	Trevor	0.5
	Difficulty	0.583

	Options	*Closeness*
a)	Hedwig	1
b)	RonWeasley	0
c)	Herminone	0
d)	Scabber	0.5
	Difficulty	0.17

In the above example, we show two sets of options for the MCQ. The difficulty levels of the options are 0.583 and 0.17 respectively. We use the same metrics defined in the previous section for calculating *Closeness* and *Difficulty*. According to those metrics, the first set of options makes the MCQ more difficult than the second set. Intuitively, since the options in the first set belong to some subclass of `Creature` class, the test takers will find it difficult to guess the correct answer (0.25 probability for guessing the correct answer). The second set contains *RonWeasley* and *Hermione* as two of the options. Since those two options can be easily categorized as wrong answers, the probability of guessing the correct will be increased to 0.5. Hence, the difficulty level of the MCQ which uses the second set of options should be less when compared to the MCQ generated with the first set.

3.2.2. MCQ generation from edge-label-set and list P_1

Algorithm 3 Distractor generating algorithm

```
1: Input: $C$ (Restriction Class of the predicate)
2: Output: $D_{complete}$ (List of possible distractors)
3: InstanceList $D_{complete} \leftarrow \phi$
4: for all instance $d$ in $O$ do
5:     if $(Disjoint(\mathcal{L}(d)) \cap \{C\} \neq \phi)$ then
6:         Add $d$ to $D_{complete}$
7:     end if
8: end for
9: return return $D_{complete}$
```

We consider the edge-label-set $\mathcal{L}$(`trevor,neville`) in Table 8 for our illustration. The list P_1 of this label-set contains the predicate `isPetOf`. This means that the $\mathcal{R}(\mathcal{L}$(`trevor`)) contains a restriction of the form $\Im$`isPetOf.`C (where C is some concept). From Table 7, we get, $\Im$`isPetOf.HogStudent` $\in$ $\mathcal{R}(\mathcal{L}$(`trevor`)). This restriction on `isPetOf` is not that strict when compared to the restriction in the previous case ($\exists!$`hasPet.Owl`). $\Im$`isPetOf.HogStudent` states that there should be at least one `isPetOf` relation from the instance `trevor`, and the range of the relation should be `HogStudent`. Therefore, when we frame a MCQ using the edge-label-set $\mathcal{L}$(`trevor,neville`), the only possible way to choose a distractor is by finding the instances in the complement of the class `HogStudent`. Algorithm 3 is designed to satisfy this requirement. Example 3 uses our example edge-label-set and the predicate `isPetOf` in MCQ generation.

Example 3. Trevor is the pet of ______.

	Options	*Closeness*
a)	NevilleLbottom	1
b)	ViktorKrum	0.415
c)	Hedwig	0
d)	Errol	0
	Difficulty	0.138

	Options	*Closeness*
a)	NevilleLbottom	1
b)	Hedwig	0
c)	Crookshanks	0
d)	Scabber	0
	Difficulty	0

In the above example, the first set of options makes the MCQ difficult than the second set of options. The *Closeness* values of some options are zero because of the fact that the Reduced-node-label-sets of such option instances cannot be related to the $\mathcal{R}nls$ of the correct answer instance (we consider those instances as unrelated ones).

4. Related Work

Papasalouros et al. (2008) suggested 11 strategies based on classes, properties and terminologies of ontologies for framing MCQs and the corresponding distracting answers. Their MCQ generation method based on these strategies lack proper theoretical backing to support when to use which strategy, and the stem of all the generated questions remains the same (*'Choose the correct sentence'*).

Cubric and Tosic (2010); M.Tosic and M.Cubric (2009) generated MCQs of knowledge level (*'Which of the following definition describes the concept C?'*), comprehension level (*'Which one of the following response pairs relates in the same way as a and b in the relation R?'*), application level (*'Which one of the following examples demonstrates the concept C?'*) and analysis level (*'Analyze the text x and decide which one of the following words is a correct replacement for the blank space in x.'*). Their method does not work if the ontology lacks proper annotation. Strategies similar to Papasalouros's strategies are adopted to find the distracting answers (distractors) for the generated MCQs.

Another MCQ generation method is by Alsubait et al. (2012). They present an approach called similarity-based approach for generating analogy type questions. In their question generation algorithm, a set of parameters are introduced to control the difficulty level of the generated questions. They argue that the difficulty level of a questions can be increased by finding the distractors which are similar to the correct answer(s). The approach which the authors illustrate is limited to analogy type questions.

In addition to the above MCQ generation approaches, a few researchers worked on rule-based methods for question answer generation. SWRL rule-based method by Zoumpatianos et al. (2011) is one such approach.

5. Comparative Study and Evaluation

In order to support the approaches presented in this paper, we implemented our techniques in Java 1.7 using Jena framework (2.11.0) as a portable library. We use Jena Ontology API[3] to explore classes and to look up restrictions in the ontologies.

Table 10: Ontologies used for evaluation

Ontology	*Classes*	*ObjectProperties*	*Individuals*
HarryPotter_book	17	5	12
People & Pet	60	15	21
Geographical Entity	34	4	306

We consider three example ontologies (i) HarryPotter_book Ontology[4] (ii) People & Pet Ontology [5] (iii) Geographical Entity Ontology [6] for our experimentation. Among the example ontologies, HarryPotter_book Ontology was developed by us and used as the example ontology throughout this paper. People & Pet Ontology is a well known ontology that is usually used for ontology development tutorials. The Geographical Entity Ontology (GEO) is an ontology of geopolitical organizations and their subdivisions. The specifications of the three ontologies are presented in Table 10. Some necessary disjointness between classes were not there in the ontologies (ii) and (iii). So, we added a few axioms to those ontologies to make them suitable to generate expected results. In GEO, we have added axioms to state that the classes: `'group of dependencies'`, `'group of major administrative subdivision'`, `nation`, `'geopolitical dependency'` and `'major administrative subdivision'` are disjoint. We added axioms to People & Pet ontology to state that `cat, duck, giraffe, person, sheep` and `tiger` are disjoint classes. We also included the axiom ($\texttt{male} \sqcap \texttt{female} \sqsubseteq \bot$), and included axioms to state that `publication, vehicle` and `animal` are disjoint concepts. Three assertions, `Male(joe)`, `Male(walt)` and `Male(fred)`, are added to generate distractors which are closely related to the correct answer (in our MCQ examples).

For a comparative study, we present the sample MCQs generated by the approaches presented in Alsubait et al. (2012); Cubric and Tosic (2010); Papasalouros et al. (2008); M.Tosic and M.Cubric (2009) from People & Pet Ontology, in Table 9. Each option is marked with (T) or (F) to represent the key and the distractor respectively. Question 1 in the table is generated using the approach proposed in Cubric and Tosic (2010); M.Tosic and M.Cubric (2009). Since the ontology lacks

[3]`http://jena.apache.org`
[4]`http://bit.ly/1sOzeoc`
[5]`http://bit.ly/1dPPLsc`
[6]`http://bit.ly/1fp9uxr`

Table 9: Sample MCQs generated from People & Pet ontology using existing approaches

1. Which of the following response pairs related in the same way as *Walt* and *Huey* in the relation *hasPet*?	
a. *Joe hasPet Fido* (T)	b. *Fred hasPet Huey* (F)
c. *Walt hasPet Louie* (F)	d. *Minne hasPet Tibbs* (F)
2. Choose the correct answer	3. Choose the correct answer.
a. *Walt hasPet Fido* (T)	a. *Fido is a dog* (T)
b. *Minne hasPet Fido* (F)	b. *Tom is a dog* (F)
c. *Minne hasPet Dewey* (F)	c. *Fluggy is a dog* (F)
d. *Kevin hasPet Louie* (F)	d. *Louie is a dog* (F)
4. Choose the correct answer.	5. *Haulage Truck Driver : Driver*
a. *Man is a Person* (T)	a. *Quality Broadsheet : Newspaper* (F)
b. *Cat is a Person* (F)	b. *Giraffe : Sheep* (F)
c. *Duck is a Person* (F)	c. *Bus : Vehicle* (T)
d. *Giraffe is a Person* (F)	d. *Giraffe : Cat Liker* (F)

proper annotations, other question types mentioned by the authors cannot be generated. Questions 2 to 4 are generated using class, proper and terminology based strategies (respectively) described in Papasalouros et al. (2008). Question 5 is an analogy type question from the approach in Alsubait et al. (2012). Example 4-8 show sample MCQs generated from the ontology (ii), using our approaches.

Table 11: Number of questions generated by the proposed approach on different ontologies

Ontology	*node-set based approach*	*edge-set based approach*
HarryPotter_book	12	8
People & Pet	21	2
Geographical Entity	305	0

Table 11 shows the count of the MCQs generated per ontology using our two approaches. We have taken $w = 3$ for our experiment. Sample questions generated from GEO ontology is given in Appendix. A detailed list of multiple choice questions generated by our approaches using the three example ontologies is given in `http://bit.ly/1lFpI6Z`.

Example 4. Choose `a person and an oldlady, having only cat as pet.`

	Options		*Closeness*
a)	Minne	(T)	1
b)	Joe	(F)	0.375
c)	Walt	(F)	0.458
d)	Fred	(F)	0.55
	Difficulty		0.461

Example 5. Minne has pet _______.

	Options		*Closeness*
a)	Tom	(T)	1
b)	Louie	(F)	0.75
c)	Fido	(F)	0.75
d)	Joe	(F)	0
	Difficulty		0.5

Example 6. Choose `a White van man and a dog owner, drives a van and a white thing, reads only tabloid, having some cat as pet.`

	Options		*Closeness*
a)	Mick	(T)	1
b)	Fred	(F)	0.392
c)	Fido	(F)	0
d)	Joe	(F)	0.547
	Difficulty		0.313

Example 7. Choose `a dog owner, having exactly one dog as pet.`

	Options		*Closeness*
a)	Joe	(T)	1
b)	Dewey	(F)	0
c)	Minne	(F)	0.375
d)	Tom	(F)	0
	Difficulty		0.125

Example 8. Joe has pet ______.

	Options		*Closeness*
a)	Fido	(T)	1
b)	Louie	(F)	0.75
c)	Tom	(F)	0.75
d)	Huey	(F)	0.75
	Difficulty		0.75

6. Conclusion and Future Work

Our approaches propose the generation techniques of two new categories of multiple choice question types based on the node-label-sets and edge-label-sets of the instances in an ontology. We introduced a technique called Label-set-Reduction to make the label sets favorable for generating MCQs by reducing their cardinality. The feasibility of the approaches are studied by experimenting them on a synthetic ontology and on two other familiar ontologies. Our experiments prove that the proposed approaches, when used with semantically-rich ontologies, can provide successful cases. While the proposed approaches work well in defining the semantics of questions, the problem of generating syntactically correct question items is only partially tackled. In order to overcome syntactic problems, more sophisticated natural language generation techniques should be utilized in future implementations of the presented approaches.

In this paper, we assumed that the difficulty level of a MCQ is purely dependent on how closely its correct answer is related to its distractors. We consider all stems with same level of importance, and assume that the options of a MCQ make one question difficult (or easy) than the other. Nevertheless, the question statement itself can make a MCQ difficult (or easy). For example, consider the stem of Example 2 *HarryPotter hasPet* _____, we can decrease the difficulty level of this question by adding the class information of the pet of Harrypotter in the stem (*HarryPotter hasPet* _____*, an Owl*).

Another observation we made during our work is about the relevance of the generated MCQ w.r.t. the domain of the ontology. For example, consider one of our previous example ontology, GEO, an ontology about geopolitical organizations and its subdivisions. It is observed that not all MCQs generated by using our approaches are relevant. Example MCQs in Appendix are MCQs generated from the GEO ontology. When we compare the stem of Example 5 (in Appendix) to the stems of Example 1 to 4 (in Appendix), Example 5 appears to be a less relevant MCQ since its stem *Choose a nation* is less related to the geopolitical organization, but more related to a generic ontology which talks about countries, continents and nations.

A simple method to find the relevance of a MCQ (generated from our node-label-set based approach) is by identifying the classes and properties which are more related to the domain. If the stem-set of an instance contain those relevant classes or properties, we can say that the MCQ generated from that instance is relevant (otherwise not relevant). In our previous example, if we identify the class `nation` as not relevant to GEO ontology, we can state that Example 5 (in Appendix) is a less relevant MCQ. An automated method for finding the relevant classes and properties in an ontology is required to enhance our current approaches. The scope of the works done by Peroni et al. (2008) and Wu et al. (2008) to identify key concepts from an ontology needs to be studied as a future endeavor.

References

Al-Yahya, M. M., 2011. Ontoque: A question generation engine for educational assesment based on domain ontologies. In: ICALT. IEEE Computer Society, pp. 393–395.

Alsubait, T., Parsia, B., Sattler, U., 2012. Mining ontologies for analogy questions: A similarity-based approach. Vol. 849 of CEUR Workshop Proceedings. OWL Experiences and Directions.

Baader, F. v., 2007. The Description logic handbook : theory, implementation, and applications. Cambridge University Press, Cambridge.

Cubric, M., Tosic, M., 2010. Towards automatic generation of e-assessment using semantic web technologies. In: Proceedings of the 2010 International Computer Assisted Assessment Conference. URL `http://hdl.handle.net/2299/4885`

Hitzler, P., Krötzsch, M., Rudolph, S., 2009. Foundations of Semantic Web Technologies. Chapman & Hall/CRC.

Horrocks, I., Sattler, U., Tobies, S., 2000. Reasoning with individuals for the description logic shiq. CoRR cs.LO/0005017.

M.Tosic, M.Cubric, 2009. SeMCQ- Protege Plugin for Automatic Ontology- Driven Multiple Choice Question Tests Generation. In: Proceedings of the 11th International Protege Conference. URL `http://protege.stanford.edu/conference/2009/poster-session.html`

Papasalouros, A., Kanaris, K., Kotis, K., 2008. Automatic Generation of Multiple Choice Questions from Domain Ontologies. International Association for Development of the Information Society, pp. 427–434.

Peroni, S., Motta, E., d'Aquin, M., 2008. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In: ASWC. pp. 242–256.

Schmidt-Schaubß, M., Smolka, G., Feb. 1991. Attributive concept descriptions with complements. Artif. Intell. 48 (1), 1–26. URL `http://dx.doi.org/10.1016/0004-3702(91)90078-X`

Wu, G., Li, J., Feng, L., Wang, K., 2008. Identifying potentially important concepts and relations in an ontology. In: International Semantic Web Conference. pp. 33–49.

Zoumpatianos, K., Papasalouros, A., Kotis, K., 2011. Automated transformation of swrl rules into multiple-choice questions. In: Murray, R. C., McCarthy, P. M. (Eds.), FLAIRS Conference. AAAI Press.

Appendix

Sample MCQs generated from Geographical Entity Ontology using node-label-set and edge-label-set based approaches are given below:

Example 1. Choose a `geopolitical dependency, a member of exactly one sovereign state.`

	Options		*Closeness*
a)	Jersey	(T)	1
b)	Illinois	(F)	0.5
c)	Maryland	(F)	0.5
d)	Hawaii	(F)	0.5
	Difficulty		0.5

Example 2. Choose a `geopolitical dependency, a member of exactly one sovereign state.`

	Options		*Closeness*
a)	Anguilla	(T)	1
b)	Massachusetts	(F)	0.5
c)	Lowa	(F)	0.5
d)	Uruguay	(F)	0
	Difficulty		0.33

Example 3. Choose a `major administrative subdivision, a part of some nation and a member of exactly one sovereign state.`

	Options		*Closeness*
a)	Florida	(T)	1
b)	Aruba	(T)	0.5
c)	Bermuda	(F)	0.5
d)	Guam	(F)	0.5
	Difficulty		0.5

Example 4. Choose a `group of major administrative subdivision.`

	Options		*Closeness*
a)	US State	(T)	1
b)	US Territories	(F)	0
c)	US Commonwealths	(F)	0
d)	Israeli-occupied Territories	(F)	0
	Difficulty		0

Example 5. Choose a `nation`.

	Options		*Closeness*
a)	India	(T)	1
b)	Bermuda	(F)	0
c)	Aruba	(F)	0
d)	Jersey	(F)	0
	Difficulty		0

Note that the difficulty levels of Example 4 and Example 5 are calculated as 0. This is because, the ontology is not semantically rich enough to relate the correct instance and the distractor instances.